CERN Summer Student report

Damian Raczkowski

Supervisors: Julien Leduc, Richard Bachmann

Summer 2024

1 Introduction

This summer, I had the privilege of participating in the Summer Student Programme at CERN, where I was selected to join the organization for 8 weeks. During this time, I contributed to the CERN Tape Archive (CTA) project, which serves as the archival storage system for the custodial copy of all physics data at CERN.

CTA is implemented as the tape back-end to EOS, making the EOS and CTA combination CERN's archival solution for the Large Hadron Collider (LHC) Run-3 and beyond. Given the critical role of CTA in preserving the vast amounts of data generated by CERN, ensuring a reliable and standardized testing process is essential.

To address this, I developed and deployed a Helm chart designed to standardize the testing process for CTA within the CI/CD pipeline. This Helm chart automates and unifies the configuration of testing environments, significantly streamlining and simplifying the process of deploying changes to CTA. Additionally, the project involved integrating this solution into the existing CI/CD pipeline, enabling continuous testing and validation of code before its deployment to the production environment.

These enhancements have made the testing process for CTA more efficient, allowing for earlier detection of potential issues and ultimately contributing to higher software quality and system stability.

In this report, I present what has been changed in the project testing environment setup.

2 Current setup

This chapter addresses the overall flaws of the existing solution to make the reader fully aware of why the migration was needed. Before implementing automation using Helm Chart, the deployment and management process for the CTA (Cern Tape Archive) application in the Kubernetes environment was carried out sequentially through bash scripts. While this approach was functional, it had significant limitations that impacted the efficiency and scalability of the deployment process.

2.1 Setup Workflow

In the previous solution, the deployment of Kubernetes resources was performed step by step according to a predefined sequence of operations. Each stage required the execution of specific bash commands that created resources such as Pods, Services, ConfigMaps, and PersistentVolumeClaims. However, one of the key challenges was transferring files and data between individual Pods.

These operations were executed using numerous kubectl cp commands, which were responsible for copying files from one Pod to another. This process required precise coordination, and any error at any stage could result in an incomplete or

inconsistent environment. Moreover, these operations were time-consuming and added complexity to the deployment process, presenting a significant challenge.

2.2 Configuration files

Another critical aspect of the complex deployment process was using nested configuration files. The bash scripts responsible for setting up the environment contained numerous references to configuration files that were deeply nested within the project's directory structure. These configurations were often specific to particular resources or stages of deployment, making their management difficult and error-prone.

Any configuration changes required manual updates across multiple files, increasing the risk of inconsistencies. Due to the lack of centralization and standardization, maintaining and updating configurations was cumbersome and prone to errors. This also complicated documentation efforts, as the actual state of the environment could differ from what was described if any changes were not properly recorded.

3 My contribution to CERN

In response to the challenges and limitations identified in the previous setup, my job was to improve it by introducing the Helm charts to the project and standardizing the configuration files application process for the project. Helm Charts offers a powerful solution to manage Kubernetes resources more efficiently, providing a templated and modular approach to deployment that addresses many of the issues inherent in the prior bash script-based process. It is a graduate project from the Cloud Native Computing Foundation (CNCF) and maintained by the Helm community.

3.1 Analysis of current setup

The first step of my project was to analyze the current setup in search of the things that could be synchronized to improve the time of the project setup. After analyzing the setup script, in which the working process has been demonstrated in Fig 2 I found that the KDC pod keytabs are manually copied from it to the other pods.

The next thing that I looked out for were the config files. After analysing docker container setup scripts I found that the multiple configuration files and authentication files such as SSS key-tabs were manually copied or file operations were performed on them, such as editing, or changing file ownership.

3.2 Proposed solution

The first thing that I did before migrating the setup to the helm charts was to change the way that the key tabs were passed to the CTA pods. Instead of manually passing them by utilizing Kubectl I have modified the script setting up the Kerberos instance to create Kubernetes secrets after generating key tabs for the chosen pods. So the last block from the Fig 2 is removed as that part is taken care of by the kdc side from now on.

The next thing I did was to design helm charts for the CI/CD pipeline. It ended up with two main charts:

- Init
- CTA

The Init chart contains configuration for Persistent Volume Claims, ConfigMaps, and pods for the Postgresql, init instances. It contains a kdc subchart that contains all of the configurations mentioned above to be able to create keytabs for the CTA project.

The CTA chart is more complex than the Init. It consists of five subcharts as shown in the graph 1. Each of them contains necessary configmaps, services, and pods to be able to run as a part of the CTA chart, but can be run independently as well.

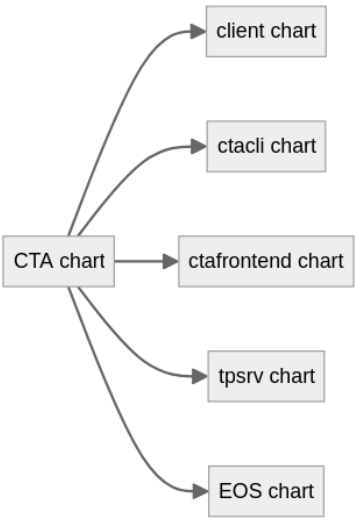

Figure 1: CTA project chart structure

Implementation of those two helm charts allowed to make the synchronization parts of the setup script as shown in fig 3. As the last step of this project, I have made a comparison of how much less time it will take for the new setup

to perform tests in comparison to the old setup. From the table below, we can see that the time to finish the singular test requires from new setup around 5% less time than with the old setup.

Table 1: Time required for the test_client.sh to finish testing

Old setup	helm charts setup
19m40 secs	18m53 secs

4 Conclusions

In conclusion, I can say that this project was very interesting and I had an amazing opportunity to get to know Helm charts better than I had before. Provided helm charts could be easily expanded by integrating the EOS chart into the CTA one which could decrease the number of dependencies to look after. Another thing that could be considered is to use the provided helm chart in production.

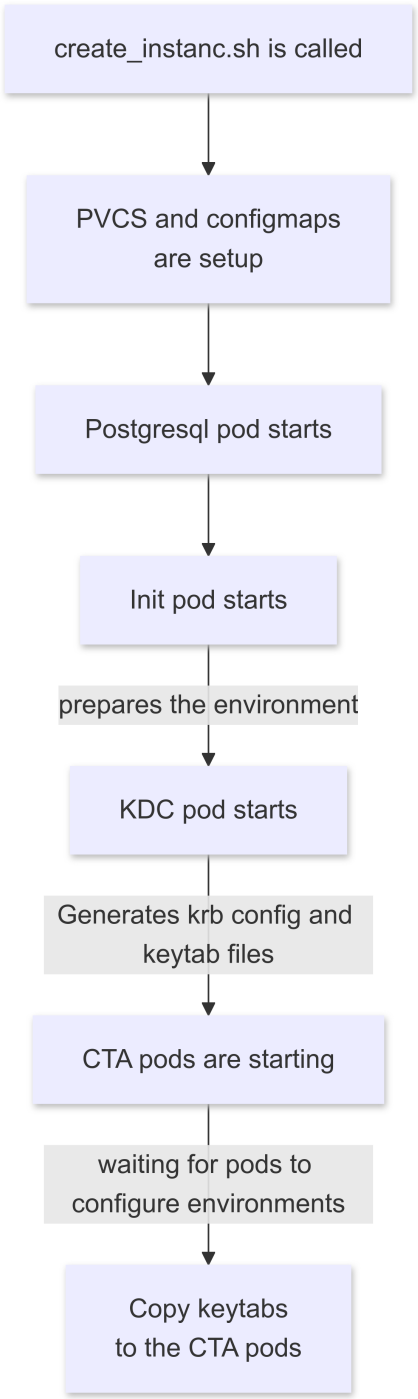

Figure 2: old setup script workflow

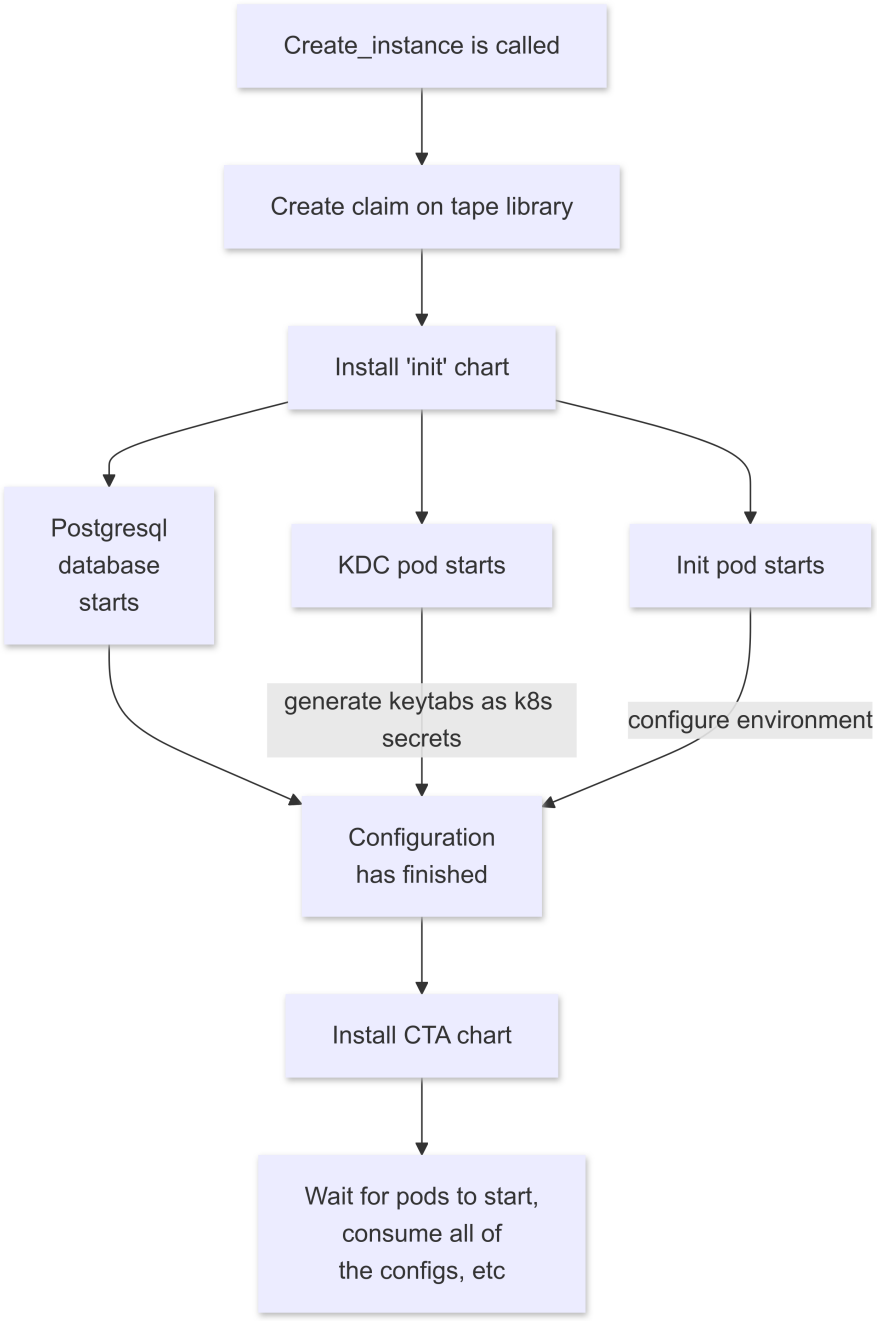

Figure 3: New setup script workflow